UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1052

Programme 1

EDITION STRUCTUREE APPROCHE HYPERTEXTE

Coopération et Complémentarité

Anne-Marie VERCOUSTRE

Juin 1989

RR_1052

Structured Editing - Hypertext Approach
Cooperation and Complementarity

Anne-Marie Vercoustre

Abstract

Hypertext systems have a great potential for writting and displaying documents and kwnoledge bases. As they are now widely available, many technical and conceptual problems have been identified. We argue here that such systems could take advantage of the proven technology of structured editors in order to provide both the user and the system with a conceptual document model as a sound basis for the hierarchical links. A prototype combining structured editing and hypertext facilities proposes two approaches to implementing non-hierarchical links to subtrees: the first one uses the *paths* from the tree root, while the second one uses tree pattern matching as a first step towards semantic and more managable links.

Edition structurée - Approche hypertexte
Coopération et complémentarité

Anne-Marie Vercoustre

Résumé

Les systèmes d'hypertextes, très en vogue actuellement, sont une approche prometteuse pour le dévelopement et la consultation de documents et de base d'informations. Toutefois de nombreux problèmes ont été identifiés. Nous pensons que de tels systèmes pouraient s'appuyer sur la technologie éprouvée des éditeurs structurés pour donner, tant au système qu'à l'utilisateur, un modèle conceptuel de document permettant d'améliorer les aspects liés aux liens hiérarchiques. Un prototype combinant les avantages de l'édition structurée et des systèmes d'hypertextes propose deux approches pour implanter les liens non-hiérarchiques vers des sous-arbres: la première utilise les *chemins d'accès* à partir de la racine, la deuxième utilise la recherche de schéma d'arbre (tree pattern matching) comme une première approche vers des liens plus sémantiques et plus gérables.

Edition structurée - Approche hypertexte Coopération et Complémentarité

Anne-Marie Vercoustre
INRIA
B.P.105, 78153 Le Chesnay, France

e-mail: vercous@inria.inria.fr

Abstract

Les systèmes d'hypertextes, très en vogue actuellement, sont une approche prometteuse pour le dévelopement et la consultation de documents et de base d'informations. Toutefois de nombreux problèmes ont été identifiés. Nous pensons que de tels systèmes pourraient s'appuyer sur la technologie éprouvée des éditeurs structurés pour donner, tant au système qu'à l'utilisateur, un modèle conceptuel de document permettant d' améliorer les aspects liés aux liens hiérarchiques. Un prototype combinant les avantages de l'édition structurée et des systèmes d'hypertextes propose deux approches pour implanter les liens non-hiérarchiques vers des sous-arbres: la première utilise les *chemins d'accès* à partir de la racine, la deuxième utilise la recherche de schéma d'arbre (tree pattern matching) comme une première approche vers des liens plus sémantiques et plus gérables.

1 Introduction

Les systèmes de production de documents peuvent être classés en plusieurs familles selon qu'ils sont centrés sur l'un des aspects suivants:

- la mise en page du document en vue de l'édition finale,
- l'écriture du document et sa manipulation au niveau logique,
- l'ensemble des documents référencés par le document (la base de documents et d'informations)

Dans la première famille on trouve, soit des outils batch (les formatteurs) comme Troff ou TEX, soit des éditeurs interactifs de type WYSIWYG[1] (Frame-

[1] What You See Is What You Get

maker, Macwrite, Macdraw). Si ces outils s'intéressent à la structure du document, c'est plus en terme de structure de page [JOL87].

Les éditeurs interactifs et structurés de la deuxième famille, comme Mentor-Rapport [MEL84], Grif [QUI86], Concerto [CON86], séparent l'aspect logique du document de sa mise en page. L'interface n'est généralement pas complétement WYSIWYG, au moins pour le découpage en page. Ces systèmes permettent aussi de manipuler des structures représentant des éléments non textuels, comme des tableaux, des formules mathématiques ou des programmes.

Les sysèmes d'hypertextes, enfin, mettent l'accent sur les relations entre différents documents et permettent de naviguer d'un document à l'autre en suivant des liens. Cette approche généralise la notion d' **index,** qui donne accès à une information située ailleurs dans le document ou dans un autre document (référence bibliographique, référence à une autre section).

L'approche hypertexte permet également de jeter les bases d'outils traitant les aspects de conception et d'écriture (qui est différent de celui de l'édition) d'un document, comme assemblage et organisation d'idées et dont la méthodologie est encore à étudier [NAN85], [BIS87].

Bien que la grande souplesse d'organisation des systèmes hypertexts soit très attrayante, le besoin de structuration apparait dans plusieurs aspects. D'une part pour la production de (gros) documents écrits qui sont généralement hiérarchiquement structurés, d'autre part pour l'affichage de vues structurées et résumées des documents, ainsi que des informations sélectionnées par le système sur requête de l'utilisateur.

De même que l'édition structurée et non-structurée peuvent être complémentaires [MEL84], nous pensons que l'édition structurée et l'approche hypertexte peuvent et devraient coopérer, pour tirer profit des deux approches.

Alors que l'édition structurée s'appuie sur une structure de référence unique et décrite formellement, les hypertextes offre une organisation plus libre de l'information (non hiérarchique) selon des critères multiples.

La description formelle permet de maintenir en permanence une organisation cohérente des documents selon certains modèles de données, tandis que les systèmes d'hypertextes permettent de gérer des références inter-documents et de naviguer dans la base d'information ainsi constituée.

La complémentarité des deux approches est démontrée par une implantation expérimentale avec le système Centaur. Nous avons réalisé une extension de l'éditeur structuré Centaur vers un système de type hypertexte, utilisant le mécanisme d'annotations du système [DON84], [CEN87]. L'implantation utilise des annotations dans le langage Lisp, comme liens typés exécutables vers des documents stockés séparément. Par ailleurs, une interface à la HyperCard [APP87],[BRA88] a été implantée: des boutons matérialisent les types de liens existants; l'utilisateur peut créer dynamiquement et interactivement ses propres liens et types de liens. Les liens sont crées en Lisp, sous forme d'arbres abstraits, éditables sous Centaur.

Après avoir rappelé briévement, en sections 2 et 3, les caractéristiques et les avantages de l'édition structurée et des hypertextes, nous présentons, en section 4, les mécanismes d'annotations et la notion de contexte utilisés dans Centaur. Dans la section 5, nous décrivons l'application hypertexte et son interface utilisateur.

Le mécanisme implanté peut être utilisé pour d'autres applications qu'un système hypertext. Nous en donnons deux exemples en section 6: d'une part un système de Help, d'autre part un système de gestions de versions ou d'archivage de documents, comme dans Neptune [DEL87] or Concerto [AND88].

2 L'approche hypertexte

La notion d'Hypertext n'est pas nouvelle puisqu'elle date de 1945 [BUS45], mais les systèmes opérationnels, liés à la disponibilité du logiciel et matériel appropriés, ne sont apparus que dans les années 70. Plus récement, HyperCard [APP87] sur Macintosh a popularisé ce type de système. Comme introduction au monde d'hypertext, et pour avoir un historique des systèmes existants, on peut se reporter à [CON87].

La principale caractéristique des systèmes d'hypertextes est de permettre la consultation d'un certain nombre de documents et d'informations en suivant des **liens.** Les liens qui permettent de se déplacer d'un document à un autre sont matérialisés graphiquement sur l'écran par un bouton ou un élément typographique spécial (comme les références dans un document écrit) qui peut généralement être activé comme un bouton.

Il y a essentiellement deux types de liens:

- **Les liens hiérarchiques** (appelés aussi liens organisationnels), permettant de voir un document plus ou moins "en profondeur", souvent associés à un mécanisme de "zoom".
- **Les liens non hiérarchiques,** utilisés pour des applications spécifiques, comme les références croisées internes ou inter-documents, ou des liens vers des "annotations" d'utilisateurs concurrents [WAL88]. On peut aussi avoir des liens pour gérer les versions, ou pour créer un ordre linéaire sur des documents.

Les liens sont généralement typés, au moins par leur nom, et visualisés à l'aide d'un "bouton", permettant à l'utilisateur de choisir le type de lien à suivre.

Outre les types de liens prédéfinis, les systèmes d'hypertextes offrent souvent à l'utilisateur la possibilité de définir ses propres liens et types de liens.

2.1 Les avantages d'Hypertext

Les avantages des sysèmes d'hypertextes sont maintenant bien connus. Ce sont principalement:

- La gestion des références internes et inter-documents et les facilités pour accéder à tous les documents ainsi reliés. L'approche hypertexte permet une lecture non linéaire d'un document, ainsi qu'une construction non hiérarchique.
- La modularité de l'information: le même texte peut être référencé à partir de plusieurs endroits et n'a donc pas besoin d'être dupliqué. Par exemple, un document pourra faire référence à une bibliographie, sans en dupliquer tout ou partie dans son document. L'information est donc toujours cohérente puisque les mises à jour ne se font qu'à un seul endroit.
- La structuration dynamique de l'information: à l'aide des liens, il est possible de réorganiser l'information, de façon hiérarchique ou non.

Les documents hypertexts sont très différents des documents sur papier. La phase qui permet de passer d'un environnement hypertext, c'est-à-dire d'une base d'informations, à un document écrit construit progressivement à partir d'une partie de ces informations, est encore un sujet d'étude, celui du processus d'écriture. La manipulation de petites unités de documents, sans structuration à priori, correspond bien à l'organisation informelle d' idées. Les liens établis dynamiquement permettent alors une structuration a posteriori qui permet l'élaboration du document.

2.2 Les inconvénients d'Hypertext

Les inconvénients d'Hypertext ont été analysés dans [CON87], [BRO88b], [BRO-88c]. Les principaux sont les suivants:

- Avec la multiplicité des liens, l'utilisateur est vite "perdu". De nombreux mécanismes ont été proposés pour donner à l'utilisateur des moyens de s'orienter : possibilité de revenir en arrière, carte du chemin parcouru, vue graphique des liens inter-documents.
- Le problème du maintient de la cohérence des liens et de l'intégrité du système est un des problèmes les plus importants à résoudre dans les systèmes d'hypertextes [BRO88]. La plupart des systèmes existants laissent la cohérence du système à la charge de l'utilisateur. Dans HyperCard, par exemple, rien n'est prévu pour supprimer les boutons accédant des cartes qui n'existent plus.
- La portée des commandes, comme par exemple la recherche de **pattern**, est mal définie: s'appliquent-elles au document courant, à l'ensemble des documents ouverts ou déjà consultés, à tous les documents liés par un type spécifique de liens, ou à l'ensemble de la base ?
- L'affichage ou l'impression d'un document hypertexte n'est pas toujours facile à spécifier, en vue de produire un document bien structuré pour la lecture. L'utilisateur devrait pouvoir spécifier la nature et l'organisation des informations concernées, comme dans le système NLS-Augment d' Engelbart [ENG68]. Une autre approche intéressante est celle d'assemblage virtuel de

documents [NAN87] qui permet de projeter la base d'informations selon un modèle de document structuré.

3 L'édition structurée

Les éditeurs de documents structurés séparent la structure logique des documents manipulés de leur affichage ou de leur impression. Une étude des principaux systèmes pour la manipulation de documents structurés peut être trouvée dans [QUI87]. Nous ne nous intéressons ici qu'aux éditeurs interactifs. On citera, par exemple, Grif [Qui87], [FUR87], Mentor-Rapport [MEL84], Concerto [AND88], Interleaf [MOR87]. Les éditeurs structurés ont été initialement développés pour une famille particulière de documents, celle des programmes. C'est le cas de Mentor [DON80], dont Mentor-Rapport est seulement une instance, du Cornell Synthesizer [TEI81] ou de Gandalf[MED82].

Ces systèmes utilisent une représentation arborescente des documents, construite selon le modèle générique de structure. Les atomes, dans cette structure, sont généralement la ligne ou le paragraphe, éléments dont l'édition est faite par un éditeur de texte classique.

Certains systèmes (Grif, Concerto) permettent de manipuler des documents contenant des références vers une autre partie du document ou vers un autre document, premier pas vers des documents non-linéaires.

Les avantages de l'édition structurée qui sont dégagés, ([NAN87] ou [FUR88]), le sont surtout par opposition à l'édition textuelle classique. Les principaux sont les suivants:

- La séparation entre la représentation interne du document et sa présentation accroit la flexibilité et la réutilisabilité.
- L'utilisation d'un modèle de présentation permet une normalisation et homogénisation des documents. De plus, cela dégage l'utilisateur des aspects présentation pendant la phase de rédaction.
- La structure des documents reflète les relations permises entre les objets qui composent le documents et contient plus d'information que le simple texte. Le maintient de la cohérence des documents est assurée par leur conformité à une description formelle. Si l'on compare à l'approche hypertexte, les liens hiérarchiques sont ici automatiquement mis à jour de façon cohérente.
- les types représentent des contraintes supplémentaires sur les objets qui permettent d' appliquer à chaque objet un traitement approprié et d'augmenter la puissance des commandes fournies à l'utilisateur. Par exemple, des vues spécialisées (graphe, tables des matières avec niveau de détail ajustable dynamiquement) peuvent être générées automatiquement à partir des spécifications. Les commandes de navigation ou de recherche peuvent utiliser la structure du document ou seulement la structure d'arbre (recherche "en largeur d'abord" qui permet de sèlectionner les éléments de plus haut niveau).

Les inconvénients de l'édition structurée ([BIS87], [NAN85]) sont les suivants:

- Le modèle hiérarchique de document est associé implicitement à un processus d'écriture "top-down". Cela est trop contraignant pour l'utilisateur et ne correspont pas au processus d'écriture: qui ne change pas de plan en cours de route ? qui n'a pas tout à coup des idées pouvant faire l'objet d'un autre ouvrage? qui n'aimerait pouvoir retrouver facilement et relire immediatement des notes prises sur le même sujet, mais dans quel papier ?
  Un réalité, la rédaction d'un document implique une grande quantité d'informations et d'autres (petits) documents, sans liens hiérarchiques entre eux.
- Les modèles de documents sont trop strictes: même s'il est possible de choisir ou de définir son propre modèle, il est souvent (trop) difficile de passer d'un modèle à un autre. Ceci est d'autant plus contraignant que le modèle de document est la première chose que les systèmes d'édition structurée demandent à l'utilisateur de choisir.

Nous avons vu que le problème des systèmes d'hypertextes est la gestion de tous les liens crées. Franz Halaz, un des développeurs de Notecards, ayant fait des statistiques sur NoteCards, indique que la structure hiérarchique est très importante dans un réseau de documents Hypertext, et que les liens de *références non hiérarchiques* sont beaucoup moins fréquents.

Il faut donc s'appuyer sur des abstractions de haut niveau pour formaliser ces liens: Les liens de type hiérarchiques dans les hypertextes seront avantageuseument remplacés par une description formelle de la structure hiérarchique du document, comme dans les éditeurs structurés, description utilisable par le système pour assurer la validité de ces liens.

4 Centaur

Centaur [CEN87] est un système générique de manipulation de documents structurés. Lorsqu'on lui fournit une description syntaxique et sémantique d'un langage, Centaur génère un environnement spécifique pour ce langage, incluant un éditeur structuré. Centaur est une version de seconde génération (en Lisp) du système Mentor [DON80] qui est utilisé depuis plusieurs années comme environnement de développement de programmes.

Mentor-Rapport [MEL84] était une première expérience d'environnement d'édition de documents techniques combinant l'édition structurée et textuelle. Cette expérience a été reprise dans Centaur-Rapport [RAZ89], prototype utilisant la même approche que Mentor-Rapport, mais réalisant une meilleure intégration avec le composeur (Latex, Troff) utilisé pour produire le document final. C'est sur Centaur-Rapport que s'appuie notre application hypertexte, mais les mécanismes utilisés sont des mécanismes généraux de Centaur, en particulier de son noyau.

Centaur s'appuie sur une représentation interne des documents sous forme arborescente, dite de *syntaxe abstraite.* La syntaxe abstraite définit les arbres légaux dans le langage édité. Le noyau de Centaur (Virtual Tree Processor ou VTP [LAN85]) définit et implémente un protocole pour supporter la création, la manipulation et le stockage des arbres de syntaxe abstraite. Il peut être vu comme une collection de types abstraits de données, correspondants aux différentes entités manipulées par le système. De plus le VTP possède des caractéristiques permettant d'assurer la modularité et l'extensibilité du système, en utilisant la programmation par objets. En particulier, le mécanisme d'annotation permet d'étendre le système par adjonction dynamique de nouveaux types d'objets.

4.1 Les annotations

La plupart des documents contiennent des références à des informations situées à un autre endroit du document ou dans un autre document (note en bas de page, renvoi à un autre chapitre, référence bibliographique). Ces informations sont des *annotations* du document qui n'interfèrent pas avec la structure du document principal.

Un mécanisme général d'annotations avait initialement été inclu dans Mentor [DON84] pour manipuler de façon structurée, les commentaires dans les programmes. Puis elles ont évolué vers un mécanisme plus général permettant d'enrichir les représentations abstraites dans des buts variés. Dans Centaur, les annotations sont organisées selon leur rôle spécifique et le formalisme dans lequel elles doivent être exprimées.

Les *annotations* sont typées par le *décor* auxquelles elles appartiennent: la définition d'un décor inclue son nom, le type des objets qui peuvent être annotés par ces annotations et le type des objets qui peuvent être utilisés comme annotation dans ce décor. Grâce au mécanisme d'annotation, Centaur permet une combinaison non hiérarchique de documents dans des formalismes différents, que l'on appellera documents multi-formalismes.

Les annotations étant fortement typées par la définition de leur décor, il est possible, dans une approche orientée objets, d'avoir une visualisation et une sélection spécifique de chaque décor.

Cette propriété est importante dans les applications de type hypertexte, où les liens sont souvent caractérisés par des boutons déclenchant une action qui dépend du type du lien.

4.2 Les contextes et le partage

Le VTP peut être vu comme une collection de types abstraits de données constituant une sorte de base de données. Dans cette base de données, il est souvent nécessaire de conserver des références sur des emplacements particuliers,

indépendamment de la valeur du sous-arbre référencé à cet endroit[2].

La notion de *contexte* matérialise l'idée d'un ou plusieurs emplacements dans l'arbre. Les sous-listes, les sous-arbres, les annotations sont des exemples de tels contextes.

La méthode habituelle pour désigner un sous-arbre d'un arbre donné utilise des pointeurs vers les noeuds de l'arbre. Cette solution est simple et facile à utiliser, et la plupart des primitives de Centaur fonctionnent en utilisant des pointeurs. Cependant, lorsqu'on veut faire des traitements plus compliqués, cette méthode n'est pas toujours adaptée.

C'est pourquoi une nouvelle méthode pour désigner les sous-arbres a été introduite dans Centaur [CLE88]. Cette méthode désigne un sous-arbre (ou un ensemble de sous-arbres) d'un arbre donné en utilisant la notion de chemin d'accès (path). Nous appelons *path* le lien allant du somment de l'arbre au sommet du sous-arbre désigné. Avec les paths, on peut considérer les relations entre deux arbres, de façon beaucoup plus indépendante de l'implémentation de ces arbres. Il est beaucoup plus facile qu'en utilisant les pointeurs, de répondre à une question comme : "ces deux sous-arbres sont-ils fils de ce troisième arbre"?

Les *paths* peuvent être utilisés pour améliorer la gestion des annotations et donc étendre leur domaine d'application:

Primitivement introduites comme une généralisation des listes de propriétés, les annotations ont tout d'abord été utilisées comme une valeur attachée au sous-arbre lui-même. Les annotations étaient, comme les commentaires, éditées et sauvegardées en même temps que l'objet annoté. Puis le besoin s'est fait sentir de considérer non seulement des valeurs d'annotation, mais également des emplacements d'annotations. Ainsi quand un objet est édité, l'annotation accrochée à l' **emplacement** de cet objet n'est pas affectée par la modification.

Les emplacements d'annotations peuvent être implantés en utilisant le système de paths. Ces paths doivent être mis à jour quand l'emplacement correspondant change. Par exemple, à la commande "inserérer avant" correspond un incrément de 1 sur tous les paths ayant, comme facteur gauche, le path associé à la position courante. Quand un emplacement d'annotation est **à l'intérieur** du sous-arbre édité, la mise à jour est un peu plus difficile, parce que le nouveau sous-arbre peut avoir une structure très différente de l'ancien. Dans ce cas, des algorithmes de *différenciation* d'arbres [FRI89] peuvent permettre d'établir la correspondance entre les anciennes positions et les nouvelles.

Ensuite, on veut pouvoir partager des annotations entre plusieurs documents. Le partage de sous-arbres, par l'intermédiaire des annotations permet de s'affranchir de la stricte structure d'arbre. La sauvegarde de ces arbres partagés, sans introduire de cycle, peut également être effectuée en utilisant une méthode de "path".

[2]Le mot "partage" signifie ici le partage d'une partie de document comme sous-unité de plusieurs autres documents. Cela correspond formellement au partage d'un sous-arbre par plusieurs arbres. Cela n'a rien à voir avec le problème d'accès à un document partagé entre plusieurs utilisateurs concurrents.

Enfin, on peut vouloir sauvegarder et accéder à une annotation indépendemment de l'objet annoté. En effet, particuliérement dans le cadre des sysèmes d'hypertextes, l'annotation est elle-même un document (ou partie de document) qui doit pouvoir être accédé, indépendemment de tout lien avec d'autres documents. Il s'agit de rendre symétrique le lien annotant-annoté.

Le problème de la sauvegarde indépendante (et cohérente) de documents partagés n'est pas encore résolu dans sa généralité. Nous en reparlerons plus loin.

4.3 Hypercentaur

Hypercentaur est un environnement multi-langages et multi-décors qui permet d'implanter facilement des applications de type hypertext. Il utilise une famille particulière de décors, ceux correspondant aux annotations en Le_Lisp[3], langage d'implantation de Centaur.

Les annotations Hypercentaur ont les caractéristiques suivantes:

- Toutes les fonctionnalités sont génériques par rapport au décor auquel elles appartiennent. Le décor sera en général caractéristique de l'application qui l'utilise. Dans les applications de type hypertext, les décors, comme les Webs de Intermedia [YAN87], permettent de séparer les liens selon le contexte de l'application.
- Il est possible de définir dynamiquement des sous-types de décor à l'intérieur d'une application; ceci permet de mieux structurer les applications, et de faciliter la création de nouveaux types de "liens" par l'utilisateur.
- Du code exécutable est attaché à chaque sous-type à l'intérieur de l'annotation. Quand un sous-type est activé (par programme ou par action de l'utilisateur), le code correspondant est sélecté par filtrage dans l'annotation (arbre Lisp).
- Des boutons peuvent être crées dynamiquement et associés à un sous-type. Quand un bouton est activé, le code correspondant à ce sous-type est activé.

```
exemple :
    (selectq arg1
         (eval-in-package
             (setq #:sys-package:colon 'interface))
         (printp (ecrirep (current-tree-k)))
     )
```

Cet exemple est tiré de l'environement Centaur-Lisp. Le code Lisp ci-dessus est le contenu de l'annotation. Le type de cette annotation a été déclarée auparavant par la déclaration de son décor comme annotation d'arbres Lisp par des arbres Lisp.

Si l'on regarde cette annotation (un *selectq* Lisp), on peut identifier deux selecteurs ("eval-in-package" et "printp"). Le code associé à "eval-in-package"

[3] Le_Lisp est une marque déposée de l'INRIA

est prévu pour être activé par programme (l'évaluateur Lisp). Le code associé à "printp" sera activé par l'utilisateur par un bouton "printp", lorsqu'il voudra l' impression de la structure interne de son arbre courant (au lieu de la vue textuelle habituelle).

5 Application hypertexte

Nous avons utilisé les mécanismes hypercentaur pour ajouter les avantages de l'approche hypertexte à ceux d'un éditeur structuré de production de documents. Nous avons défini un décor spécifique, appelé "hyperlink", pour implanter les liens non hiérarchiques. Un tel décor "hyperlink" doit être défini pour tous les langages des documents sources (documents dans le langage Rapport ou dans n'importe quel langage implanté sous le système, comme Ada ou Lisp).

5.1 Annotation en Lisp

Les noms des liens sont implantés comme sous-types du type "hyperlink" qui caractérise cette application. Un lien spécifique est activé en évaluant le code Lisp associé au sous-type correspondant.

exemple:

```
(selectq arg1
     (source (#:interface:voir "/u/croap/vercous/rapport/hypertext/"
                               "hyperlink"
                               'all))
     (help (#:interface:voir "/u/croap/vercous/rapport/hypertext/"
                             "hyperhelp"
                             '((3 (1 . all)))))
     (notes (#:interface:voir "/u/croap/vercous/rapport/hypertext/"
                              "abstract"
                              'all)))
```

Ici, on a trois liens vers des documents externes implantés par une annotation dans le décor "hyperlink" avec les sous-types "source", "notes", "help". A chacun de ces liens est associé la fonction Lisp (#:interface:voir). L'activation de ce lien provoquera le positionnement sur le document voulu, après l'avoir chargé et affiché dans une nouvelle fenêtre, s'il n'y était pas déjà.

all et ((3 (1 . all))) sont des exemples de *paths* qui désignent respectivement le sommet de l'arbre et un sous-arbre particulier (troisième fils de la racine, puis premier élément de la liste, implantées sous forme d'arbre binaire).

Les noms des liens renseignent l'utilisateur sur le type de document qui est relié à cet endroit.

Contrairement à HyperCard, nous n'utilisant pas un langage ad-hoc pour implanter les liens. N'importe quel code Lisp peut être mis dans les liens et activés par bouton. Toutefois des fonctions spécifiques (comme la fonction :voir), ont été écrites pour cette application et des appels à ces fonctions sont générés automatiquement par l'interface interactive.

5.2 Interface

L'interface utilisateur, à la HyperCard, permet de:

- créer un lien spécifique (i.e d'un nom donné), de l'objet courant vers un objet (arbre ou sous-arbre) désigné par son nom, ou sa localisation à l'aide de la souris. Comme dans HyperCard, la création interactive du lien génère du code, Lisp jouant ici le rôle d'Hypertalk.
- créer un nouveau type de lien dont l'utilisateur donne le nom. Le lien est d'abord créé sans référencer aucun objet, sauf pour certains liens prédéfinis qui peuvent être automatiquement connectés à des documents prédéfinis.
- visualiser un lien: la visualisation est orientée objet. Par défaut, le lien est visualisé par son nom, mais on peut redéfinir une autre méthode. En particulier, on peut visualiser tout ou partie de l'objet référencé par le lien.
- détruire un lien.
- éditer un lien, en tant qu'arbre Lisp, sous Centaur-Lisp.
- activer un lien par bouton; chaque type de lien peut être matérialisé par un bouton créé dynamiquement et visualisant le nom du lien.

  Exemple de visualisation de lien:

```
[notes]
[help]
[CEN86]
```

Les liens "notes" et "help" sont visualisés par leur nom (affichage par défaut), tandis que le lien de référence bibliographique (bib) est visualisé par la référence bibliographique adressée.

Dans la version actuelle, les "souches" des liens (i.e [notes] ou [help]) ne sont pas directement utilisés comme boutons. Ils pourront l'être dans la nouvelle version de Centaur où la sélection d'une annotation sera orientée par l'objet, c'est-à-dire que l'interprétation de la sélection pourra dépendre du type de l'objet sélecté.

5.3 Discussion

Avec ce prototype, nous voulions résoudre les points suivants:

- Référence à des objets structurés partagés, sans introduire de cycle. Nous avons éliminé les cycles possibles par une indirection à travers une annotation. L'objet partagé peut être référencé en utilisant un "chemin d'accès" (path).

- Référence à des objets persistants extérieurs: comme dans les autres systèmes hypertextes, notre prototype implante le lien vers ces objets sous forme d'appel à la fonction de chargement de ces objets. Dans ce cas, il n'y a pas de problème de mise à jour des liens, puisqu'on établit un lien sur "une racine" d'objet. Il s'agit *seulement* d'être cohérent avec le système de gestion de versions des objets du système (le lien doit-il relier la dernière version de l'objet ou une version particulière ?)
- Référence à des objets persistants et partagés par d'autres documents. Cette situation demande a être expliquée un peu plus en détail.

Prenons l'exemple d'un document faisant référence à un élément de bibliographie, celle-ci étant un document séparé pouvant êre partagé par un grand nombre de documents. Le problème est que la bibliographie peut être modifiée de façon indépendante des documents (alors que ceux-ci ne sont pas en mémoire). Se pose alors le maintient de la cohérence de ces références bibliographiques, qui doit se faire en gérant les *références entrantes,* c'est-à-dire les liens vers des sous-objets (subtrees). Les références entrantes sont généralement gérées par un système d'identificateurs attachés à tous les éléments exportables. Comme le nombre de tels identificateurs risque d'être très grand, les paths paraissent un meilleur moyen de représentation les liens entrants, car ils sont un moyen d'identification **relatif** à un document plus grand, qui a deja un nom.

Toutefois les mécanismes de gestion et de mise à jour de ces paths n' étant pas encore implantés, nous avons essayé une autre approche, en introduisant la sémantique du lien: la référence à la section 3 d'un document signifie seulement que la section indiquée est la troisième, au moment où le document est lu sur papier (la référence linéaire est la plus pratique pour un support écrit). Ce qui nous intéresse en réalité est la section intitulée "L'édition structurée", même si elle devient la section 2 ou 4, dans une autre version du document.

La solution adoptée est de créer un lien exécutable sous forme d'un appel à une fonction de recherche (:find). Celle-ci effectue une recherche par "pattern-matching", sur un schéma d'arbre incomplet, représentant une section d'intitulée "L'édition structurée".

Dans ce cas le lien est implanté sous la forme suivante:

```
(selectq arg1
    (ref (#:interface:find "/u/croap/vercous/rapport/hypertext/"
            "mydocument"
            (#:interface:section "L'edition structuree")))
```

On établit le lien sur un sous-objet (une section) qui est cherché ici par "pattern matching" au lieu d'utiliser le chemin d'accès. Cet exemple simple montre que, avec cette approche, il n'y a pas réellement de différence entre un lien "explicite" (aller à tel endroit, dans tel objet) et un "lien implicite", selon la terminologie de Brown [BRO88a]. L'exemple de lien implicite est celui

donné par la commande Find qui établit dynamiquement un lien sur un objet répondant à un certain critère (recherche d'un certaine chaine dans un texte, ou d'un schéma d'arbre). Dans le cas où la commande Find est interprétée comme "chercher l'occurrence suivante", on aura même un lien sur un ensemble d'objets.

En introduisant des critères sémantiques plus sophistiqués, et en prenant avantage de la description formelle des objets manipulés sous Centaur (à la fois syntaxique et sémantique), on se donnera les moyens d'implanter des systèmes hypertexts dont on saura vérifier et maintenir la cohérence.

6 Autres applications

L'utilisation de liens exécutables vers des documents structurés permet de traiter de façon uniforme de nombreuses applications .

Tout d'abord, on peut remarquer qu'il est facile d'implanter, avec cette approche, une interface traitant la hiérarchie dans les documents sous forme de liens à la Hypertext: il suffit de créer les liens "père", "fils", "frère" sous forme d'appels aux fonctions de navigation du système. Ceci met en évidence l'avantage, pour ce type de liens, de l'approche structurée qui peut utiliser la description formelle du document pour faire la mise à jour de ces liens hiérarchiques.

Cette application n'étant pas d'un intêret pratique évident, nous donnons deux autres exemples d'applications plus significatives.

6.1 Système d'aide

On peut, dans une première approche, classer les systèmes d'aide (Help) en deux catégories:

- Ceux qui, sur demande de l'utilisateur, délivrent un message, parfois plusieurs si l'utilisateur en demande encore.
- Ceux qui permettent à l'utilisateur de naviguer dans la documentation, à partir d'une position selectée par le système.

Dans les deux cas, le système doit être capable de déterminer le message ou la partie de documentation à délivrer. Cela peut être fait en utilisant un critère de recherche (mot clef, synonyme, par exemple) ou en attachant un message à chaque objet. La différence vient plus sur la possibilité donnée à l'utilisateur de continuer à explorer la documentation, si la réponse obtenue ne le satisfait pas (complétement).

La possibilité de navigation dans la documentation (browsing) est appréciée des utilisateurs, qui n'obtiennent pas toujours une réponse satisfaisante du système à des questions souvent mal formulées.

L'intérêt de cette fonctionnalité a motivé l'adaptation en sysèmes d'hypertextes, de systèmes de help existants, comme le manuel Unix*, habituellement affiché en utilisant les commandes *man* ou *more* [BRO88a].

L'utilisation de liens (calculés) vers une documentation structurée donne une architecture puissante pour réaliser un système d'aide présentant les caractéristiques suivantes:

- visualisation de la partie de documentation sélectionnée par le critère de recherche. La sélection peut prendre avantage de la structure arborescente (par exemple en sélectant en "largeur d'abord") et de la description formelle du document.
- élargissement automatique de la recherche avec les mêmes critères, ou d'autres (par exemple utilisation de synonymes).
- navigation libre dans toute la documentation, et même dans les sources du système, si les liens appropriés ont été établis, à partir (et vers) la documentation correspondante.
- édition sur papier de la documentation par l'éditeur structuré, puisque la documentation est dans un format éditable.

6.2 Gestion de versions et archivage

L'archivage de documents ou partie de documents, permettant la gestion des évolutions, doivent faire partie d'un système de production de documents.

Par ailleurs, le système doit être capable de manipuler des documents répartis en plusieurs fichiers, afin de manipuler des unités moins grosses. Les parties stockées séparément ne sont chargées (de façon transparente pour l'utilisateur) que lorsqu'elles sont effectivement accédées. Un tel système d'archivage et de séparation de document est implanté de façon industrielle dans Concerto [AND88].

Notre implantation des liens, orientée par objets, permet de choisir la vision donnée à l'utilisateur: on peut, soit présenter le document comme s'il était une seule unité (comme dans Concerto), soit opter pour une interface à la hypertext, où un bouton (stub) indiquant la place de l'objet séparé permet, à la demande, de le visualiser dans une autre fenêtre.

7 Conclusions

La réalisation du prototype Hypercentaur à partir des mécanismes d'annotations et de paths de Centaur, a fait la preuve que le VTP pouvait être utilisé comme une véritable base de données de documents structurés. La généricité des méchanismes utilisés permet de l'utiliser pour différentes applications, comme un système de gestion de versions, comme dans Neptune ou Concerto , ou un système d'aide intéractif.

Dans le contexte de l'édition de documents, l'intégration des avantages de l'édition structurée et de l'approche hypertexte ont été mis en évidence. Il s'agit, dans un premier temps, de remplacer les unités textuelles, généralement petites, apparaissant dans les noeuds des systèmes d'hypertextes, par des objets ou sous-objets structurés décrits formellement. L'utilisation de modèles formels de documents donne au systèmes des bases solides pour gérer les liens hiérarchiques, à la fois pour assurer leur mise à jour et pour construire des vues structurées de l'information. De plus, ces modèles sont une aide cognitive pour l'utilisateur qui peut naviguer dans les documents à un haut niveau de structures.

Les liens non-hiérarchiques ont été implantés soit comme une fonction "goto" utilisant les "chemins d'accès", soit comme une fonction de recherche de modèle d'arbre (pattern matching).

Continuant dans cette direction, les liens doivent être sémantiquement définis et interprétés par le système lorsqu'ils sont activés. Les liens sémantiques utilisant une logique appropriée amélioreront le contrôle et la cohérence des systèmes d'hypertextes. On peut aussi envisager d'implanter les liens en utilisant, à la place de Lisp, un langage (interprétable) de requête dans les bases de données. Ceci augmenterait la puissance de recherche dans la base de documents.

Par ailleurs, il faut étendre aux documents hypertextes, la notion de vues sur un document (structuré), obtenues par des formattages différents, en étendant la puissance des langages de décompilation comme PPML[BOR88], ou en généralisant la notion de documents conceptuels [NAN 87], qui permet de définir une correspondance (a posteriori) entre (une partie de) un document hypertexte et le document structuré visualisé ou imprimé.

8 Remerciements

Je remercie F.Rouaix pour les fructueuses discussions que nous avons eues, et pour avoir implanté la plupart des fonctionnalités hypercentaur.

Je remercie également A.Bisseret qui a inspiré en grande partie ce travail, en m'initiant avec enthousiasme à HyperCard.

References

[AND 88] E.André, A.Conchon, Un atelier Concerto pour la gestion des NEF, *L'Echo des RECHERCHES N.134*, Déc. 88.

[APP 87] Apple Computer Inc., *Macintosh HyperCard user's guide*, 1987.

[BIS 87] A.Bisseret, *Towards computer-aided text production*, Rapport de Recherche, INRIA No.665, Mai 1987.

[BOR 88] P.Borras, "PPML:Reference Manual and Compiler Implementation", *GIPE, Third Annual Review Report*, 1988.

[BRA 88] P.Brandeis, J.Kertesz, *Le livre d'HyperCard*, Editions P.S.I., 1988.

[BRO 88a] P.J.Brown, Converting Help Systems to Hypertext, *Software-Practice and experience*, vol.18(2),163-1.

[BRO 88b] P.J.Brown, Linking and searching within hypertext, *Electronic publishing*, vol.1(1),45-53, April 1988.

[BRO 88c] P.J.Brown, Hypertext: The Way Forward CACM, vol.31(7),183-191, 1988.

[BUS 45] V.Bush, "As we may think", *Atlantic Monthly*, Vol.176, p.101-108, 1945.

[CENT 87] P.Borras, D.Clément, Th.Despeyroux, J.Incerpi, G.Kahn, B.Lang, V.Pascual, *"CENTAUR : The system"*, Rapport de Recherche INRIA 777, Décembre 1987.

[CLE 88] D.Clément, *Designating subtrees in Centaur*, GIPE, Esprit project N0.348, January 88.

[CON 86] *L'atelier Concerto*, Journées techniques, Perros-Guirrec 4,5,6 février 1986.

[CON 87] J.Conklin, Hypertext: An Introduction and Survey, *IEEE Computer*, Vol.20, No.19, Sept.1987.

[DEL87] N. M. Delisle, M.D.Shcwartz, Contexts, A partitioning Concept for Hypertext, *ACM transactions on Office Information Systems*, Vol.5(2), April 1987.

[DON 80] V.Donzeau-Gouge et al., *Programming environments based on structured editors, The Mentor experience*, Rapport de Recherche INRIA, n.26, 1980.

[DON 84] V.Donzeau-Gouge, G.Kahn, B.Lang, B.Mélèse, Document structure and modularity in mentor, *Proceedings of the ACM Software engeneering Symposium on Practical software development environments*, Sotware Engeneering notes Vol. 9 - no 3, 1984.

[ENG 68] D.C. Engelbart, W.K. English, A Research Center for Augmenting Human Intellect, *AFIPS Conf. Proc.*, Vol.33, Part 1, The Thomson Book Company, 1968.

[FRI 89] F. Frilley, *Differenciation d'ensembles structurés*, Thèse de Doctorat, Université Paris VII, Février 1989.

[FUR 87] R.Furuta, Concepts and Models for Structured documents, *Actes des Journées "Structure de/for Documents"*, p 9-38, Aussois , Janvier 1987.

[FUR 88] R.Furuta, V.Quint, J.André, Interactively editing structured documents, *Electronic Publishing*, vol.1(1),19-44, April 1988.

[JOL 87] V.Joloboff, Représentation des documents: Etat de l'art, recherche, *Actes des Journées "Structure de/for Documents"* , p 81-103, Aussois, Janvier 1987.

[GAR 88] Pankaj K.Garg, Abstraction mechanisms in hypertext In *Communications of ACM*, Vol.31, No.7, July 1988.

[LAN 85] B.Lang, *Mentor. Design and implementation of the kernel of a program manipulation system*, Conférence on integrated project support environment, J.A.McDermid Ed,York, 1985.

[MED 82] R. Medina-Mora, *Syntax directed editing: toward integreted programming environments*, CMU-CS-82-113, Carnegie-Mellon University, 1982.

[MEL 84] B.Mélèse, Structured editing - Unstructured editing, Coopération and complémentarity, *Actes de 2ième Colloque de génie logiciel*, Nice Juin 1984.

[MOR 87] R.A. Morris, The Interleaf User Interface, in PROTEXT III: *Proceedings of the Third Internationnal Conference on Text Processing System*, J.J.H.Miller ed.,p.20-29, Boole Press,1987.

[NAN 85] M.Nanard, L.Nanard, J.Falgueirettes, *Top down or Bottom up Approach for Document Structuration*, In PROTEXT I, Dublin, October 1985.

[NAN 87] M.Nanard, L.Nanard, Interactive Manipulation of Structured Documents, In PROTEXT IV, Boston, October 1987.

[NAN 88] M.Nanard, H.Richy, L.Nanard, Conceptual documents - Mecanisms for specifying active views, *ACM Conference on Document Processing Systems*, Santa Fe, Dec. 1988.

[QUI 86] V.Quint, I.Vatton, Grif:An Interactive System for Structured Document manipulation, In *Proceeding of EP'86 Conférence*, Univ. of Nottingham,14-16 April 1986.

[QUI 87] V.Quint, *Une approche de l'édition structurée des documents*, Thèse d'Etat, Univ. Sci. Tech. et Méd. de Grenoble, mai 1987.

[QUI 87] V.Quint, Les systèmes pour la manipulation de documents structurés, *Actes des Journées "Structure de/for Documents"*, p 39-80, Aussois, Janvier 1987.

[RAZ 89] P.Razac, *Manipulation de documents sous Centaur*, Mémoire d'ingénieur, CNAM, à paraître, 1989.

[RIJ 86] C.J. Rijsbergen, A non-classical logic for information retrieval, *The Computer Journal*, Vol.29(6), p.29-33, 1986.

[SMI 86] J.B. Smith et al., *WE:A Writing Environment for Professionnals*, Technical Report,Univ. of North Carolina at Chaptel Hill, August 1986.

[TEI 81] T. Teitelbaum, T. Reps, The Cornell program Synthesizer: a syntax directed programming environment, *Communication of the ACM*, Vol 24(9), pp. 563-573, 1981.

[WAL 88] J.H. Walker, Supporting Document development with Concordia, *IEEE Computer*, p.48-59, January 1988.

[YAN 87] N. Yankelovitch, G.P. Landow, D. Cody, Creating Hypermedia Materials for English Literature Students, SIGCUE, spring/summer 1987.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique